University of Ibadan
Journal of Science and Logics in ICT Research

Solving Travelling Salesman Problem Using an Improved Ant Colony Optimization Algorithm

[1]✉RUFAI, K. I., [2]USMAN, O. L., [3]OLUSANYA, O. O. and [4]ADEDEJI, O. B.

[1,2,3,4]Computer Science Department, Tai Solarin University of Education, Ijagun, Ogun State, Nigeria.
[2]Research Centre for Cyber Security, Faculty of Information Science and Technology, University Kebangsaan Malaysia, Bangi, Selangor, Malaysia.
Emails of authors: rufaiki@tasued.edu.ng; usmanol@tasued.edu.ng; olusanya_oo@tasued.edu.ng; adedejiob@tasued.edu.ng

Abstract
Travelling Salesman Problem (TSP) is a well-known and mostly researched problem in the field of combinatorial optimization. In this study, an attempt was made to model an improved Ant Colony Optimization (ACO) algorithm that has a fast convergence speed and very good global optimization ability when solving TSP based on Nigeria52 dataset. The proposed improved ACO algorithm was tested using the MATLAB (R2018a) software, with TSP conceived as a complete weighted Hamiltonian cycle of 52 cities, and with Abuja as the starting point of the salesman's tour. The results of an improved ACO simulation show that the algorithm was able to improve convergence by avoiding local minima with an optimal path length (cost) of 5866.9249 when the value of exploitation (α) is less than that of exploration (β). The results also considerable improvement compared against tradition ACO algorithm with 98% precision.

Keywords: *Travelling Salesman Problem, Ant Colony Optimization, Convergence, Nigria52 dataset, Combinatorial optimization*

1. INTRODUCTION

The travelling salesman problem (TSP) is a popular, and well-studied problem in the field of combinatorial optimization that draws the attention of computer scientists, mathematicians, and other experts [1, 2, 3]. Its statement appears straightforward, but it is one of the sufficiently difficult practical problems in operational research. It is also an optimization problem in terms of determining the shortest closed tour that visits all the given cities. It is referred to as a classical NP-complete problem because it has extremely large search spaces and it is extremely difficult to solve.

Most area researchers failed to find out different technique for solving the problem, first algorithm used for given problem was linear programming formulation [4]. The branch and bound algorithm used to solve TSP supports fewer than 100 cities; additionally, as the number of cities increases from 10 to 100, the branch and bound algorithm fails to provide the optimized path due to the algorithm's complexity [5, 6]. Dynamic programming was a more efficient technique than branch and bound for solving the TSP, but the dynamic programming algorithm did not support cities larger than 100. The complexity of the dynamic programming method is higher than that of the branch and bound lower bound, but for the upper bound, this algorithm fails to outperform other methods.

Dorigo et al. [7, 8] proposed ant colony optimization (ACO) in 1991 as a novel nature-inspired meta-heuristic solution to difficult combinatorial optimization (CO) problems. The ACO is a type of meta-heuristic, which are approximate algorithms used to find good enough solutions to difficult CO problems in a reasonable amount of time [7, 8]. ACO is a probabilistic technique for solving computational problems that can be reduced to finding food paths through a graph. It is

RUFAI K. I., USMAN O. L., OLUSANYA O. O. and ADEDEJI O. B. (2021). Solving Travelling Salesman Problem using an Improved Ant Colony Optimization Algorithm. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR),* Vol. 6 No. 2, pp. 158-170

inspired by the behaviour of natural ants foraging for food and returning to their nest at the end of their search, usually via a shorter route [9, 10, 11]. This algorithm belongs to the class of swarm intelligence algorithms. ACO attempts to solve this problem by utilizing a population of ants to construct tours that allow them to move from one city to another on the graph and by applying pheromone trails to the arcs that connect the nodes (cities). The ants can use the pheromone trail information to create effective TSP solutions.

The TSP can be defined as an undirected weighted graph, with cities serving as the graph's vertices, paths serving as the graph's edges, and a path's distance serving as the edge's length. It is a minimization problem that begins and ends at a specified vertex (node) after visiting each other vertex only once. There are many conventional graph-based algorithms for finding the shortest path, the most common of which are Bellman-Ford and Dijkstra's algorithms; however, due to their slow computational capabilities and routing of a single vertex more than once, these algorithms could not produce optimal solutions to TSP, thereby violating the requirement of TSP [12, 13]. Meanwhile, improving the convergence speed of swarm intelligent algorithms for optimal performance is still a work in progress [14], hence the need for this effort.

The TSP is formally defined based on the graph shown in Figure 1, where the red dots represent the set of cities and the blue lines connecting them represent the roads which connect them:

TSP Definition: Find the shortest path through a series of cities, visiting each only once.

Input: A map of cities, roads connecting cities, and distances between cities.

Output: A road sequence that will take a salesman through each city on the map, ensuring that he visits each city exactly once and travels the shortest total distance.

The problem outlined above can be approached by simply following the four steps below:

i. Using a mathematical model, determine the maximum number of tours applicable to the TSP.
ii. Make a list of every possible path.
iii. Determine the length (distance or cost) of each tour.
iv. Choose the shortest tour that provides the best/optimal solution.

Keeping the above steps in mind, we mathematically model and simulate the behaviour of virtual ants in this paper to solve TSP, using the Nigeria52 dataset, which includes latitude and longitude data as well as maps of 52 different cities in Nigeria. Furthermore, exploitation, exploration, and pheromone trails are critical parameters that must be properly adjusted to improve the ACO algorithm's convergence [14]. The study experimentally adjusts these parameters in order to achieve the above while modelling TSP solutions using the improved ACO. Therefore, the objective of this paper is to simulate the optimal cost a salesman incurs while touring 52 major cities in Nigeria using an improved ACO algorithm.

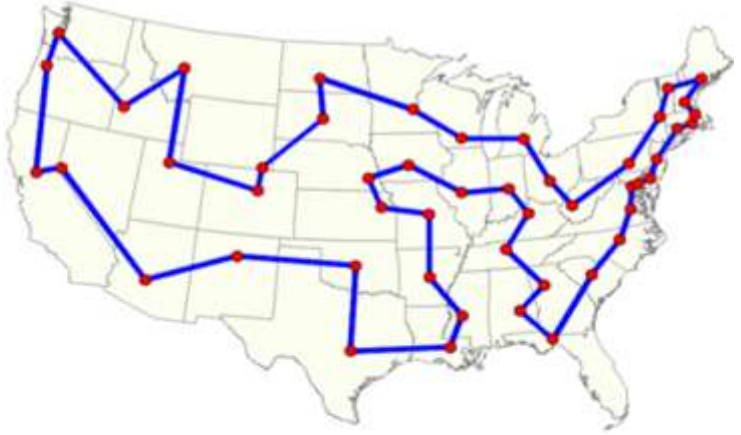


Figure 1. Map of USA showing a set of cities (red dots) and roads connecting them (blue lines)

2. RELATED WORKS

In the last few decades, a number of methods for solving TSP have been developed, based on optimization algorithms and data from various sources, with the goal of predicting optimal solutions in terms of time, distance, and cost. These methods include, but are not limited to, the genetic algorithm (GA), the evolutionary algorithm, particle swam optimization (PSO), the innovative search, ACO, and bee colony optimization (BCO) algorithm.

Moon *et. al.* [15] present a study that uses a GA to solve TSP with precedence constraints. In this study, they used topological sort to determine which vertices should be visited by the salesperson. They also provide a new crossover operator that is similar to the natural moon in many ways, such as the implementation of half-moon and full-moon.

This crossover operator chooses a random subset from the population and combines it with the chosen parents to produce offspring. The authors compared their newly developed moon crossover (MX) operator to earlier operators such as order crossover (OX) and position-based (PX) operators, discovering that their performance is nearly equivalent, but the OX and PX operators do not provide optimal results for the trials. Their approach is much more efficient for small and medium-sized problems; however, for larger problems, it provides the best solution, but there is no guarantee of optimality. One significant limitation of their study is that the topological sort they proposed for selecting vertices only works when the graph does not contain any cycles.

Tsai *et. al.* [16] proposed the heterogeneous selection evolutionary algorithm (HeSEA) for solving the large TSP. They first investigated the strengths and limitations of several well-known genetic operators, as well as local search methods for TSPs based on solution qualities and mechanisms for preserving and adding edges.

Shi *et. al.* [17] presented a novel PSO algorithm for solving TSP. When compared to the dominant algorithms for solving TSP using swarm intelligence, an undetermined searching approach and a crossover elimination method were used to increase the speed of convergence. It has been demonstrated that the presented algorithm can solve problems of large size. Furthermore, the generalized chromosome technique was used to extend the algorithm even further.

Li *et. al.* [18] proposed an improved ACO method for solving the TSP. In the study, they proposed a selection mechanism based on the Held-Karp (HK) lower bound to find the best path for TSP. It obtains information from pheromone deposition as well as heuristic information and employs the HK method to determine the best route.

Bifan *et. al.* [19] proposed ACO algorithm in which new ants remember the best solution presented thus far. Ants with Memory (AwM) was the name given to the model that was presented. They did some important work in this study, such as introducing previous knowledge of the TSP, ant system (AS), and ant colony system (ACS), defining the parameters used, explaining about AwM and merging them in ACS, and then the results were obtained by modified ants and comparing the efficiency of each algorithm. This algorithm can quickly combine into at least a proximate optimal solution. The presented algorithm is both simple and effective. This algorithm is suitable for small and medium-sized problems; however, for larger problems, it may trap in local optima.

Hlaing and Khine [20] proposed a system based on the basic ACO algorithm to improve performance in solving TSP using strategies such as sound distribution and information entropy, which were then applied to the configuration policy to modify the prior informational parameter. Also, Hlaing and Khine [20] further proposed an improved ACO algorithm that addresses the problems of high cost and solution traps in local optima. To find the best solution for TSP, two features are used: a candidate set strategy and a dynamically updated rule for heuristic parameter based on entropy. A number of preferred nodes are stored in a static list and in a dynamic candidate list. When an ant moves from one node to another, it chooses the node that is in the preferred list. This strategy is used to set up an ACS's searching scheme on larger data sets. The improvement was based on entropy and the convergence of a solution.

Stutzle *et al.* [21] conducted an experiment on computation time, taking different parameter values which can boost the performance of an algorithm in two ways - prescheduled and based on search progress. The two most common themes in evolutionary algorithms are parameter control and parameter. According to Geng *et. al.* [22], improved adaptive Simulated Annealing with greedy search was used for the solution of TSP, and they came up with three different mutation strategies for the generation of new solutions.

Hingrajiya *et. al.* [23] proposed a novel approach to ACO enhancement. In the study, they proposed an even distribution of initial ants, with at least one ant at each node. This expands the search space for solutions and increases the likelihood of finding the best results. Initially, the ants use heuristic information to choose their route. In this case, they use a large number of heuristic parameters to reduce the effect of pheromones, allowing ants to choose other paths in generating solutions. It encourages them to choose closer cities, which means they are more likely to travel along small edges. The study depicts a study for preventing stagnation and premature convergence by using an even distribution of initial ants.

According to Singh and Narayan [24], the BCO is used to solve the TSP due to its basic mechanism of bee foraging behaviour and its efficiency in determining the shortest path among various routes. When exploitation is desired, a neighbourhood search is useful. It can be used after each bee cycle to improve the quality of the solutions. Guo *et. al.* [25] investigated and evaluated several ACO algorithms that use various improving methods, and then synthesize two types of strategies (an improvement on solution construction and an update of pheromone trails) from them. They investigate and demonstrate the effectiveness of performance and application of the two strategies through experiments.

Panwar and Gupta [26] proposed an improved GA for TSP. In their research, they use the Euclidean formula to compute the distances between various cities visited and create a matrix from the data collected. They operate on a symmetric TSP, which means that the distance between two cities is the same in both directions when moving from city A to city B and vice versa. They generated a random population and then assigned them a fitness value, which is used to calculate the distance between the cities. Following that, they used the tournament selection method to select the best population from the given set, and they used the two-point crossover method to solve the TSP by combining knowledge from heuristic methods and GA. Finally, interchange mutation was used to create a new population. It appears to find better solutions for symmetric TSP, but not for asymmetric problems.

Cui and Han [27] describe the foundation of the ACO using the TSP problem, its model, benefits, and drawbacks. The existing ACO understanding process was used in solving TSP, and the simulation results show that ACO works better than others, with advantages such as fast and high precision of convergence speed, strong robustness, and many others. However, this is only possible if the number of ants is small, and the search time will be longer if there are more ants.

In their research, Khatter and Gosawmi [28] concluded that the GA can be used to solve the TSP. Depending on how the problem is encoded and which types of crossover and mutation methods are used, the GA finds the best solution for the TSP. Wei [29] selects the best parameter combination by combining the efficient selection technique. Use the improved ACO in conjunction with the best preservation policy ant system, the max-min ant system, ant-based sorting systems, and the best-worst ant system. Then, using the same TSP and parameters, perform the performance evaluation. The results of the experiment show that the planned method of parameter combination significantly improves the rate of convergence.

Gunduz *et al.* [9] proposed a new hierarchical method for solving the TSP based on swarm intelligence algorithms. In the hierarchical method, the swarm intelligence algorithms implemented combined the optimality efficiency of ACO with the time efficiency of artificial bee colony (ABC) algorithm. ACO was specifically used to provide a better initial solution for the ABC, which used the path improvement technique to achieve near-optimal results. In terms of processing time, the authors claimed that combining ACO and ABC produces better quality solutions than either approach alone. Jiang [30] proposed an ABC optimization algorithm to solve TSP in another study and demonstrated that the algorithm can efficiently and quickly find optimal or suboptimal solutions.

Asmar *et al.* [31] attempt to conduct a comparative study using various ACO algorithms. The outcomes clearly show that AS rank provides the best solutions for the entire specified target due to its ability to travel around and successfully utilize the solution space. They also compare the performance of ACO algorithms on the TSP to that of other meta-heuristics such as GRASP, Tabu Search, GA, and simulated annealing.

Hertono *et al.* [1] proposed three modifications to the TSP method based on the ACO, PSO, and 3-Opt algorithms. The ACO was used to find the optimal solution to TSP while the PSO was used to determine the best value of parameters used by ACO. The 3-Opt was used at each iteration to reduce the total tour length obtained by ACO from the feasible, entire, and different solutions. The results of implementing the three modifications revealed that only the second and third modifications provide satisfactory solutions, though the second converges at a slower rate than the third.

To compensate for the slow convergence and low efficiency of traditional ACO in solving TSP, Gao [32] proposed a new ACO algorithm that allows for the ants' search space to be expanded and potential solutions to be diversified. To diversify the solution space, a strategy of combining pairs of searching ants was used, while a threshold constant was introduced to reduce the influence of having a limited number of ants meeting. The results of a comparison with 16 state-of-the-art algorithms show that the proposed algorithm is a highly suitable method for solving the TSP, and its performance is superior to that of most algorithms.

Qamar *et al.* [2] present a novel TSP-solving algorithm based on the Best-Worst Ant System (BWAS). To improve the display of the TSP arrangement, the BWAS algorithm incorporated arrangement as a high-level type of ACO and PSO-ACO. The results for TSP arrangement show that initial trail setup for the best particle can result in a considerable shortening of the accumulated process of optimization. The mathematical test results demonstrate the viability of the proposed computation over regular ACO and PSO-ACO-based methods.

Consequently, this study proposed the implementation of an improved ACO algorithm for solving TSP based on the strengths and weaknesses of existing CO algorithms for solving TSP and the need to further improve the convergence of existing ACO-based methods. TSP was envisioned as a graph of well-connected nodes (i.e., cities) with coordinate x and y representing their latitude and longitude and using parameter tuning to determine the shortest optimal path distance.

3. METHODOLOGY

The ACO was inspired by ant foraging behaviour. The ants' behaviour is controlled by two main parameters: Alpha (α), which is the attractiveness of the pheromone to the ant, and Beta (β), which is the ant's exploration capability. If α is very large, the pheromones left by previous ants in a particular path will be deemed very attractive, causing the majority of the ants to divert their paths toward only one route, a concept known as *exploitation*. If β, on the other hand, is large, ants are more self-sufficient in determining the best path, a concept known as *exploration*. The illustrations above explain how the traditional ACO presented in Algorithm 1 (shown in Figure 2) works.

Figure 3 depicts the flowchart of an improved ACO modelling paradigm for solving TSP. According to Figure 3, an improved ACO for TSP searches for the best solution by placing each ant on the first node of the TSP graph. Initialize parameters such as exploitation (α), exploration (β), and pheromone trails (ρ), then incrementally construct solution space by applying a state transition rule to each ant to build a solution. Pheromone trails are continuously updated until all ants have constructed a complete solution. While the salesman tours the number of cities until the last city, the number of iterations is updated accordingly to obtain the best solution. TSP can be solved in graph theory by locating the Hamiltonian cycle with the smallest sum of weights for a given complete weighted graph. A TSP can be represented by a complete weighted graph *G= (N, E)*, where *N* is the set of cities (nodes) and *E* is the set of edges (paths) fully connecting all cities. Each edge $(i, j) \in E$ is assigned a cost d_{ij}, which is the distance between cities *i* and *j*. The d_{ij}, can be defined in the Euclidean space given in Equation (1). Algorithm 2 (Figure 4) depicting an improved ACO for solving TSP is coined from Figure 3.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

The improved ACO implementation employs the AS variant, in which the movement from node *i* to node *j* is defined as follows by Equation (2):

$$\rho_{ij}^{k}(t) = \begin{cases} \dfrac{r_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum \mu \epsilon \mathrm{N}_i^k(t) r_{i\mu}^{\alpha}(t)\eta_{i\mu}^{\beta}(t)}, if\ j \in \mathrm{N}_i^k(t) \\ 0, if\ j \notin \mathrm{N}_i^k(t) \end{cases} \quad (2)$$

where $\eta_{ij} = {}^{1}/_{d_{ij}}$ and α, β ≥ 0

ρ_{ij}^{k} = represents the choice probability of ant *k* transferred from position *i* to position *j*.

d_{ij} = distance between cities/nodes *i* and *j*

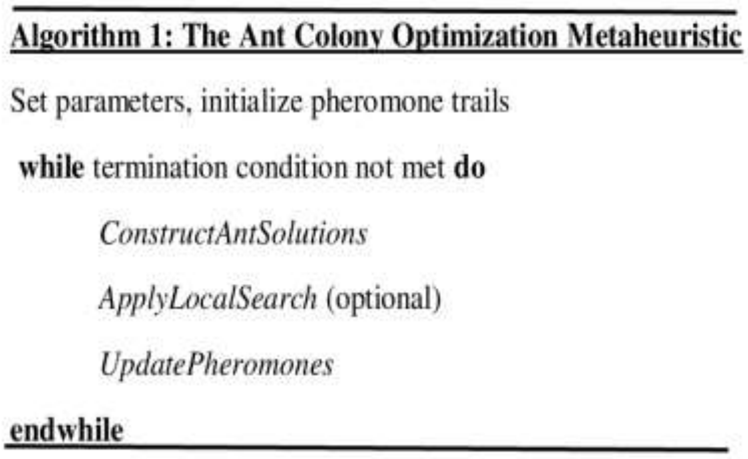
τ_{ij} = intensity of pheromone trail between cities/nodes *i* and *j*

η_{ij} = representing the heuristic information to travel to the city *j* from the city *i*, i.e., the closer the city, stronger the wish to visit it.

$\mathcal{N}_i^{\kappa}$ = set of nodes connected to point *i*, without the last visited point before *i*,

α = parameter that decides the relative control or weight of the pheromone trail.

β = parameter to show visibility when selecting the route.



Algorithm 1: The Ant Colony Optimization Metaheuristic

```
Set parameters, initialize pheromone trails
while termination condition not met do
        ConstructAntSolutions
        ApplyLocalSearch (optional)
        UpdatePheromones
endwhile
```

Figure 2. Traditional ACO algorithm. Source: Dorigo *et al.* [7].

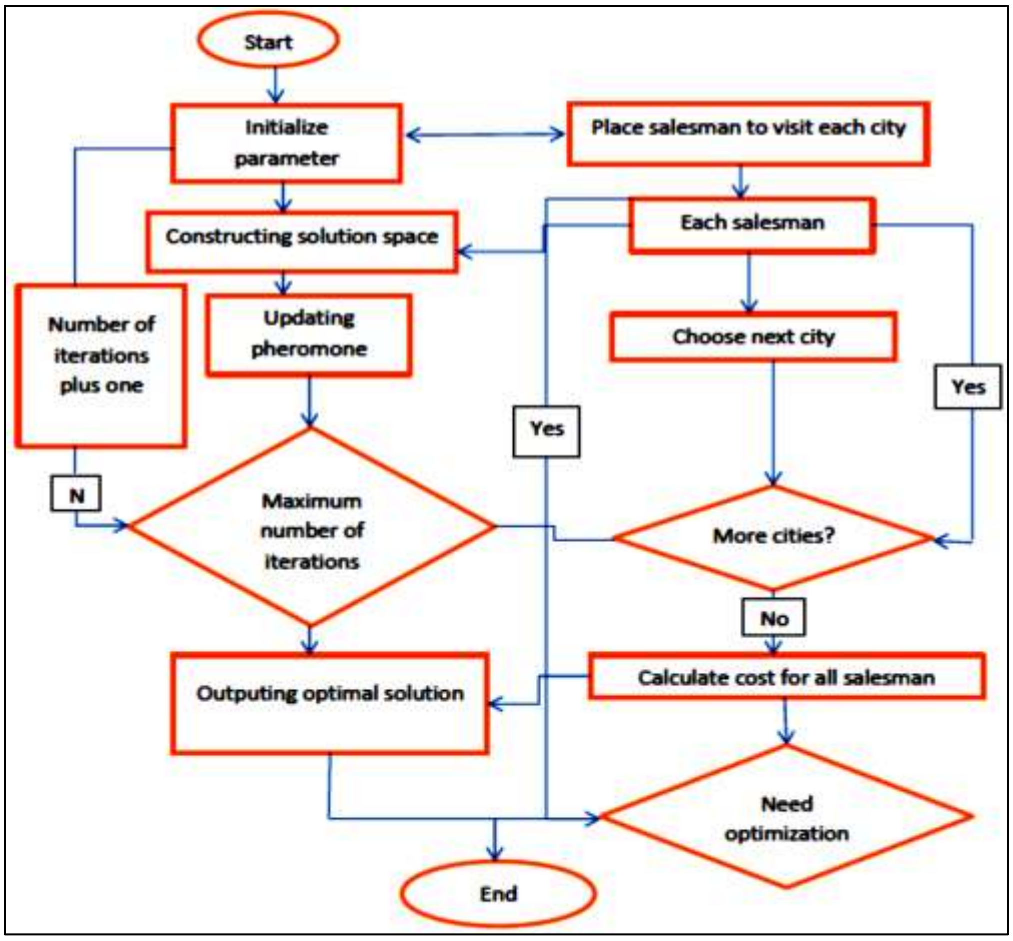

Start
Initialize parameter
Place salesman to visit each city
Each salesman
Constructing solution space
Number of iterations plus one
Updating pheromone
Choose next city
Yes
Yes
N
Maximum number of iterations
More cities?
No
Outputing optimal solution
Calculate cost for all salesman
Need optimization
End

Figure 3. Modelling of an improved ACO for solving TSP.

Algorithm 2: Improved Ant Colony Optimization for TSP

```
Set parameters, initialize pheromone trails
do Loop // Loop at this level is called an iteration
        Position each ant on a starting node of TSP Graph
        do Loop // Loop at this level is called a step
                ApplyStateTransitionRule to each ant
                IncrementallyConstructAntSolution
                Appy local UpdatePheromone rule
                Test for OptimalSolution
        while solution is not complete
        Apply global UpdatePheromone rule
while termination condition not met
        end-do Loop
end-do Loop
```

Figure 4. An improve ACO algorithm for TSP.

4. EXPERIMENTAL SETUP, RESULTS AND DISCUSSION

4. 1 Experimental Setup

The proposed improved ACO algorithm was tested using MATLAB (R2018a) software on a GPU-based processor with a processing speed of 2.70 GHz and 8.00 GB of RAM. In our case, the TSP is conceived as a problem of finding the shortest tour distance given a list of cities represented by their *x* and *y* coordinates, each of which is visited only once. The Nigeria52 dataset, which maps 52 different cities in Nigeria, was used in this study. Nigeria52 dataset was created using the Nigeria Cities Database, which can be found online at https://simplemaps.com/data/ng-cities. The original dataset's attributes include a city's latitude, longitude, state, and other variables of interest. Appendix 1 contains a list of cities and their coordinates (latitude and longitude). The implementation was carried out through a process of parameter turning during the experiment on different values of alpha (α) which represents exploitation, beta (β) which represents exploration, and pheromone (ρ) in order to balance the exploitation-exploration tradeoff. Throughout the simulation, a maximum iteration rate of 500 epochs was maintained, with a maximum colony size of 52. For all simulation runs, the Federal Capital Territory, Abuja (Node 37), is set as the starting point for the salesman's tour.

4.2 Experimental Results and Discussion

Table 1 shows the minimum costs (distances) obtained for five different simulations at different exploitation (α), exploration (β), and pheromone (ρ) values. According to Table 1, the optimal solution for an improved ACO algorithm simulated for solving TSP in this experiment based on the Nigeria52 dataset has a minimum path length (cost) of 5866.9249, as shown in Simulation_1 and Simulation_5 when α = 15 and β = 20, respectively. A change in the value of has no effect on the optimal cost. The improved ACO exhibits fast convergence speed and very good global optimization ability when the above parameters are tuned, thereby avoiding the situation's local convergence. When α = 20 and β = 15, the minimum cost is 6573.5154 (Simulation_2), while when α = β = 15, the minimum cost is 6113.4710 and 6743.0167 (Simulation_3 and Simulation_4), respectively. All our experiments show that the improved ACO algorithm performs optimally when dealing with the TSP. Figure 5-9 provide graphical illustrations of the simulation results shown in Table 1.

By attempting to solve the TSP, the proposed improved ACO algorithm achieved 98% precision when compared to the typical traditional ACO. The results demonstrate that the proposed algorithm can achieve a better solution with greater accuracy and less effort. The findings of this study are consistent with the cutting-edge solutions proposed by Gao [32] and Qamar *et al.* [2]. The implication of the above is that, for the 52 simulated cities, the proposed improved ACO algorithm can find much shorter path lengths.

Table 1. Simulated optimal cost by improved ACO algorithm

Simulation_id	**α**	**β**	**ρ**	**Minimum cost (d_{ij})**
Simulation_1	15	20	0.15	5866.9249
Simulation_2	20	15	0.15	6573.5154
Simulation_2	15	15	0.08	6113.4710
Simulation_4	15	15	0.01	6743.0167
Simulation_5	15	30	0.01	5866.9249

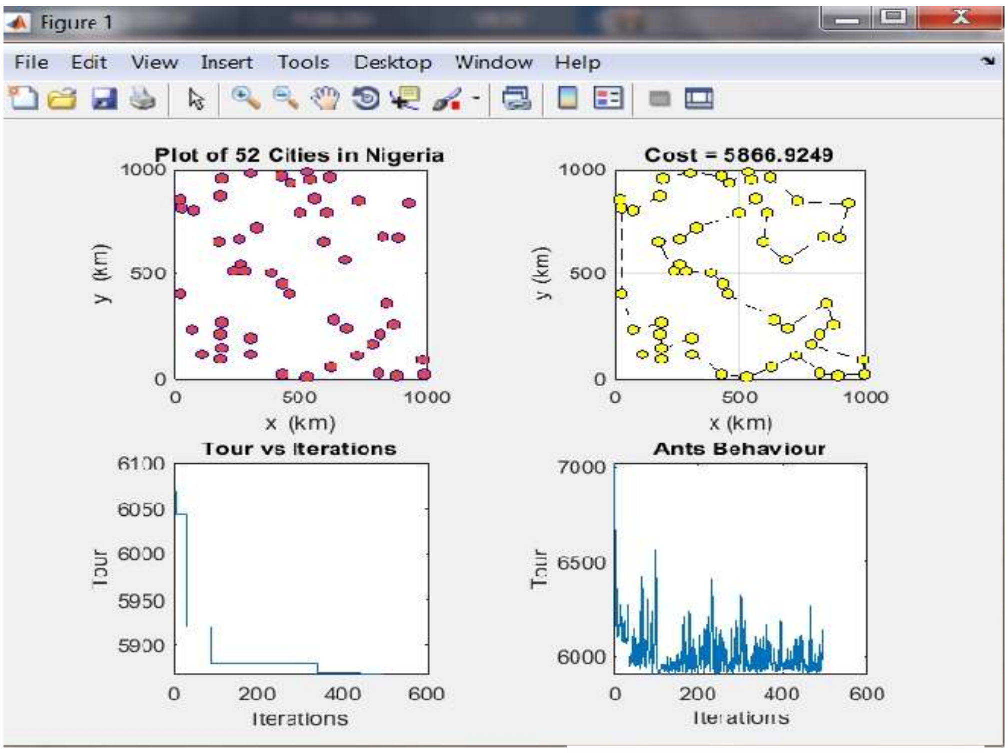


Figure 5. Graphical illustration of Simulation_1 result. Parameters: α = 15, β = 20, ρ= 0.15.

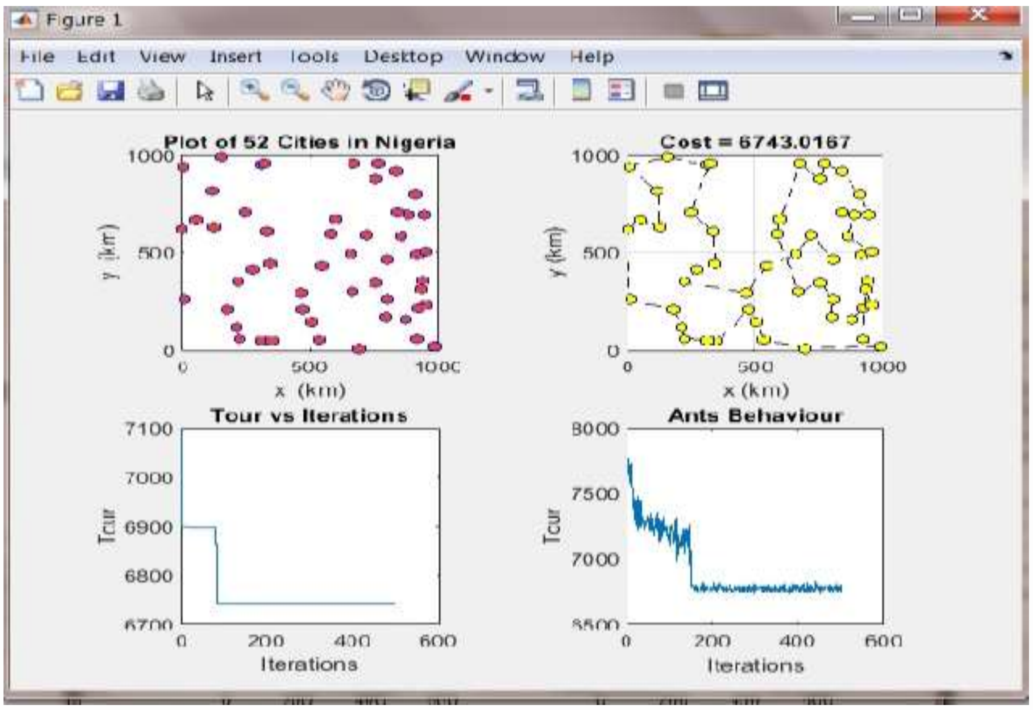


Figure 6. Graphical illustration of Simulation_2 result. Parameters: α = 20, β = 15, ρ= 0.15

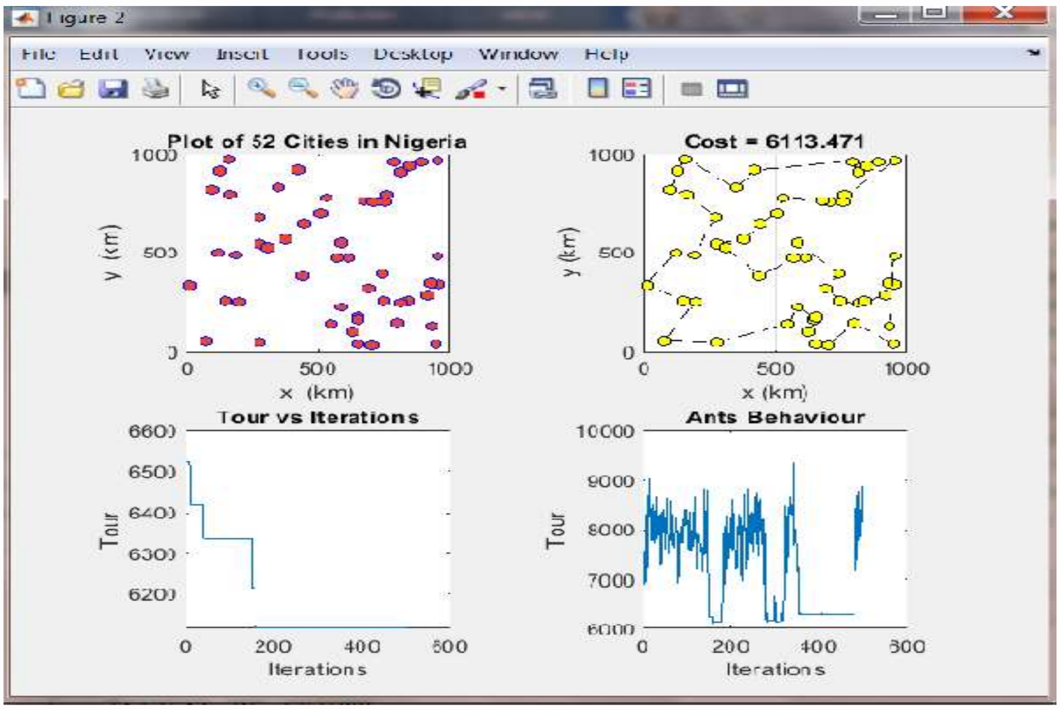


Figure 7. Graphical illustration of Simulation_3 result. Parameters: α = 15, β = 15, ρ= 0.08.

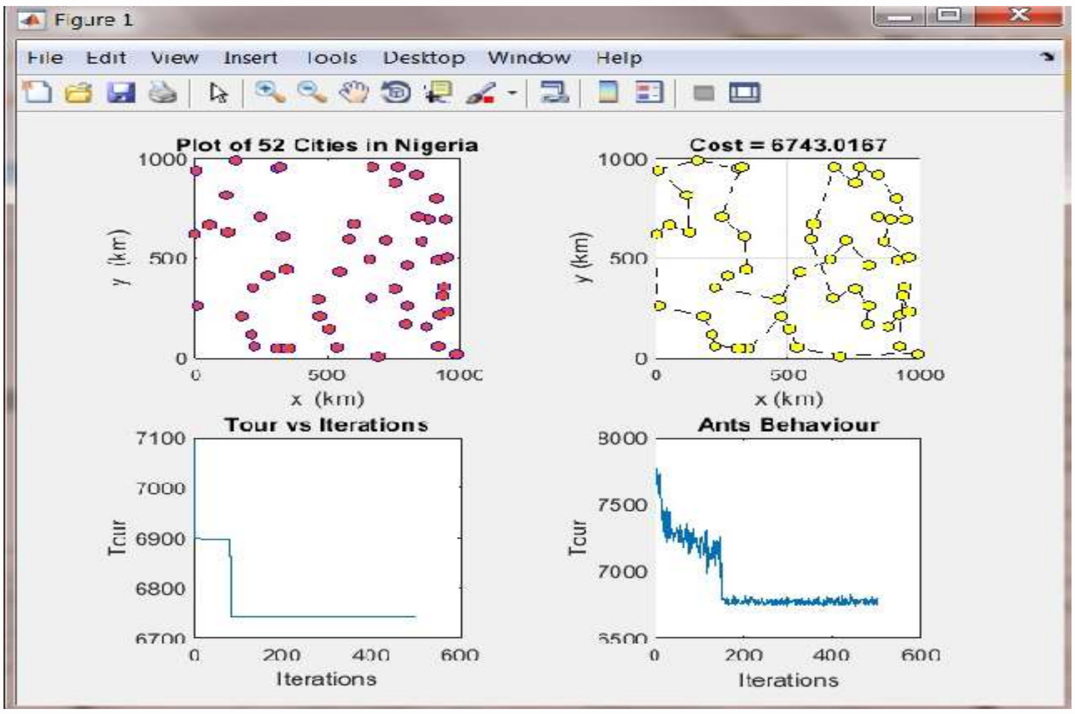


Figure 8. Graphical illustration of Simulation_4 result. Parameters: α = 15, β = 15, ρ= 0.01.

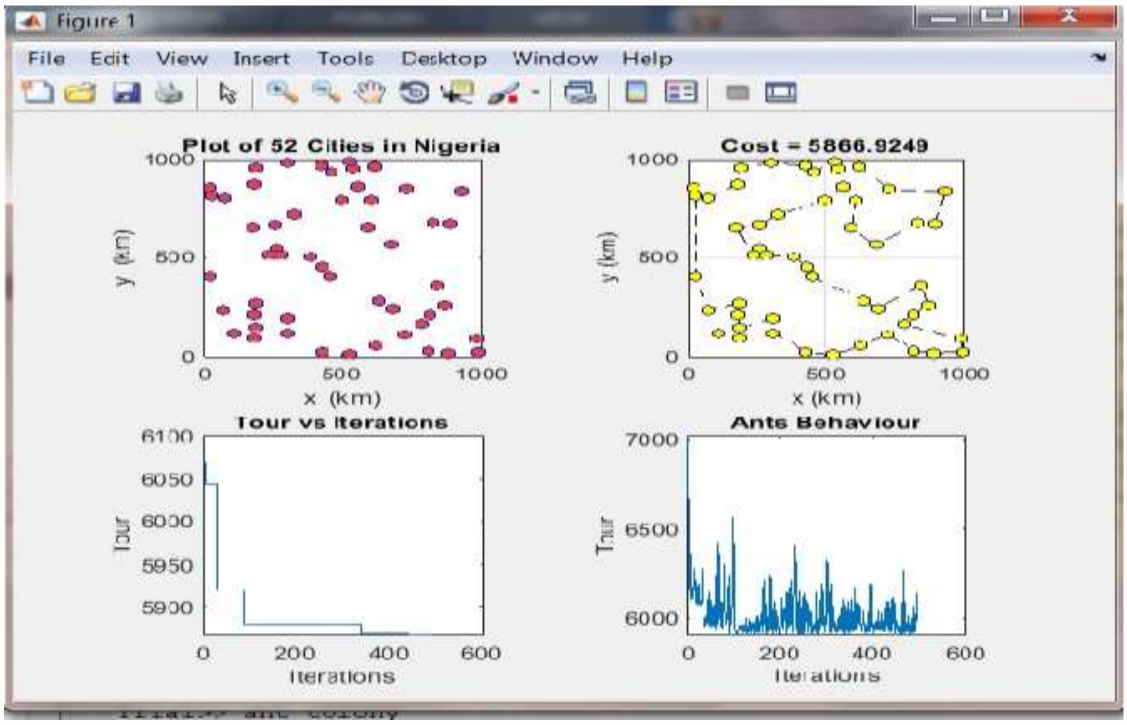


Figure 9. Graphical illustration of Simulation_5 result. Parameters: α = 15, β = 20, ρ= 0.01.

5. Conclusion

In this study, an attempt has been made to proffer optimal solution to an age-long intractable TSP using an improved ACO algorithm and Nigeria52 dataset. ACO is a type of meta-heuristic, which are approximate algorithms used to solve difficult CO problems in a reasonable amount of time. ACO is a probabilistic technique for solving computational problems that can be reduced to locating food paths in a graph. It is based on the behaviour of natural ants foraging for food and returning to their nest at the end of their journey, which is usually via a shorter route. Through the process of parameter tuning, our newly introduced improved ACO simulated in this study exhibits fast convergence speed and very good global optimization ability, achieving an optimal distance of 5866.9249 and 98% precision. Future research should attempt to implement the proposed ACO algorithm on larger datasets and compare its performance to that of other optimization algorithms.

References

[1] G. F. Hertono, Ubadah, and B. D. Handari (2018). "The modification of hybrid method of ant colony optimization, particle swarm optimization and 3-OPT algorithm in traveling salesman problem," *J. Phys. Conf. Ser.*, vol. 974, no. 1, pp. 1–7, doi: 10.1088/1742-6596/974/1/012032.

[2] M. S. Qamar *et al* (2021)., "Improvement of traveling salesman problem solution using hybrid algorithm based on best-worst ant system and particle swarm optimization," *Appl. Sci.*, vol. 11, no. 4780, pp. 1–16, doi: 10.3390/app11114780.

[3] T. Saenphon, S. Phimoltares, and C. Lursinsap (2014). "Combining new Fast Opposite Gradient Search with Ant Colony Optimization for solving travelling salesman problem," *Eng. Appl. Artif. Intell.*, vol. 35, pp. 324–334, 2014, doi: 10.1016/j.engappai. 06.026.

[4] C. Chekuri and K. Quanrud (2018). "Fast Approximations for Metric-TSP via Linear Programming," *arXiv*:1802.01242v1.

[5] Y. Li, K. Ma, and J. Zhang (2013). "An efficient multicore based parallel computing approach for TSP problems," in *Proceedings - 2013 9th International Conference on Semantics, Knowledge and Grids*, SKG 2013, 2013, pp. 98–104, doi: 10.1109/SKG. 41.

[6] F. Carrabs, R. Cerulli, and M. G. Speranza (2013). “A branch-and-bound algorithm for the double travelling salesman problem with two stacks,” *Networks*, vol. 61, no. 1, pp. 58–75, doi: 10.1002/net.21468.

[7] M. Dorigo, M. Birattari, and T. Stiitzle (2006). “Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique,” *IEEE Computational Intelligence*, no. November, pp. 28–39.

[8] M. Dorigo and C. Blum (2005). “Ant colony optimization theory: A survey,” *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, doi: 10.1016/j.tcs.2005.05.020.

[9] M. Gündüz, M. S. Kiran, and E. Özceylan (2015). “A hierarchic approach based on swarm intelligence to solve the traveling salesman problem,” *Turkish J. Electr. Eng. Comput. Sci.*, vol. 23, no. 1, pp. 103–117, doi: 10.3906/elk-1210-147.

[10] G. Bhardwaj and M. Pandey (2014). “Parallel Implementation of Travelling Salesman Problem using Ant Colony Optimization,” *Int. J. Comput. Appl. Technol. Res.*, vol. 3, no. 6, pp. 385–389, doi: 10.7753/ijcatr0306.1015.

[11] W. Qinghong, Z. Ying, and M. Zongmin (2010). “Progresses of ant colony optimization algorithm applications,” in ICACTE 2010 - 2010 *3rd International Conference on Advanced Computer Theory and Engineering, Proceedings,* 2010, vol. 1, pp. 113–117, doi: 10.1109/ICACTE. 5579052.

[12] O. L. Usman, A. O. Adebare, K. I. Rufai, and P. A. Idowu (2018). “Determination of the Shortest Path of a Nigerian University Map Using Dijkstra’S Algorithm,” *Ife J. Inf. Commun. Technol.*, vol. 3, pp. 38–48.

[13] A. O. Adebare, O. Folorunso, and O. L. Usman (2016). “Developing a Neuro-fuzzy Model for Determining Shortest Routing Path in a Computer Network,” *J. Comput. Sci. its Appl.*, vol. 23, no. 2, pp. 53–64.

[14] K. I. Rufai, O. L. Usman, R. C. Muniyandi, and L. O. A. Oyinkanola (2021). “Modelling Credit Card Payment Fraud Detection System For Financial Institutions In Nigeria Using An Improved Firefly Algorithm,” *Int. J. Inf. Process. Commun.*, vol. 11, no. 1, pp. 9–25.

[15] C. Moon, J. Kim, G. Choi, and Y. Seo (2002). “An efficient genetic algorithm for the traveling salesman problem with precedence constraints,” *Eur. J. Oper. Res.*, vol. 140, pp. 606–617, doi: 10.1007/978-3-642-16388-3_12.

[16] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao (2004). “An evolutionary algorithm for large traveling salesman problems,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 34, no. 4, pp. 1718–1729, doi: 10.1109/TSMCB.2004.828283.

[17] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and Q. X. Wang (2007). “Particle swarm optimization-based algorithms for TSP and generalized TSP,” *Inf. Process. Lett.*, vol. 103, no. 5, pp. 169–176, doi: 10.1016/j.ipl.2007.03.010.

[18] L. Li, S. Ju, and Y. Zhang (2008). “Improved ant colony optimization for the traveling salesman problem,” *Proc. - Int. Conf. Intell. Comput. Technol*. Autom. ICICTA 2008, vol. 1, no. 9, pp. 76–80, doi: 10.1109/ICICTA.2008.265.

[19] L. Bifan, W. Lipo, and S. Wu (2008). “Ant colony optimization for the traveling salesman problem based on ants with memory,” in *4th International Conference on Natural Computation*, ICNC 2008, vol. 7, pp. 496–501, doi: 10.1109/ICNC.2008.354.

[20] Z. C. S. S. Hlaing and M. A. Khine (2011). “Solving traveling salesman problem by using improved ant colony optimization algorithm,” *Int. J. Inf. Educ. Technol.*, vol. 1, no. 5, pp. 404–409.

[21] T. Stutzle *et al*. (2011). “Parameter Adaptation in Ant Colony Optimization,” in *Autonomous Search*, Y. Hamadi, Ed. Berlin Heidelberg: Springer-Verlag, pp. 191–215.

[22] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao (2011). “Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search,” *Appl. Soft Comput.*, vol. 11, pp. 3680–3689, doi: 10.1016/j.asoc.2011.01.039.

[23] K. A. Hingrajiya, R. K. Gupta, and G. S. Chandel (2012). “An ant colony optimization algorithm for multiple travelling salesman problem,” *Int. J. Sci. Res. Publ.*, vol. 2, no. 8, pp. 293–298, doi: 10.1109/ICICIC.2006.40.

[24] A. Singh and D. Narayan (2012). “Augmentation of Travelling Salesman Problem using Bee Colony Optimization,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 1, no. 2, pp. 61–65.

[25] P. Guo, Z. Liu, and L. Zhu (2012). “Improved strategies of ant colony optimization algorithms,” *Commun. Comput. Inf. Sci.*, vol. 308 CCIS, no. PART 2, pp. 396–403, doi: 10.1007/978-3-642-34041-3_56.

[26] P. Panwar and S. Gupta (2013). “Brief Survey of Soft Computing Techniques Used for Optimization of TSP,” *Int. J. Comput. Sci. Appl.*, vol. 2, no. 1, pp. 54–59.

[27] S. Cui and S. Han (2013) “Ant colony algorithm and its application in solving the traveling salesman problem,” in *3rd International Conference on Instrumentation and Measurement, Computer, Communication and Control*, IMCCC 2013, pp. 1200–1203, doi: 10.1109/IMCCC.2013.266.

[28] S. Khattar and P. Gosawmi (2014). “An Efficient Solution of Travelling Salesman Problem Using Genetic Algorithm,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 5, pp. 656–660.

[29] X. Wei (2014). “Parameters Analysis for Basic Ant Colony Optimization Algorithm in TSP,” *Int. J. u- e-Service, Sci. Technol.*, vol. 7, no. 4, pp. 159–170, doi: 10.14257 /ijunesst. 2014.7.4.16.

[30] H. Jiang (2015). “Artificial Bee Colony algorithm for Traveling Salesman Problem,” in *4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering* (ICMMCCE 2015), pp. 468–472, doi: 10.2991/icmmcce-15.2015.94.

[31] D. C. Asmar, A. Elshamli, and S. Areibi (2017). “A Comparative Assessment of ACO Algorithms Within a TSP Environment,” in *Dynamics of Continuous Discrete and Impulsive Systems-Series B-Applications & Algorithms*, vol. 1, pp. 462–467.

[32] W. Gao (2020) “New ant colony optimization algorithm for the traveling salesman problem,” *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, pp. 44–55, doi: 10.2991/ijcis.d.200117.001.

Appendix 1

List of Cities and their Coordinates.

S/N	City	Latitude	Longitude
1	Umuahia	29.68	40.60
2	Yola	12.59	43.40
3	Uyo	20.27	34.09
4	Akwa	37.90	20.21
5	Bauchi	37.15	37.97
6	Yenagoa	36.30	3.50
7	Makurdi	1.50	17.00
8	Maiduguri	59.99	60.00
9	Calabar	32.15	37.02
10	Asaba	53.66	54.73
11	Abakaliki	29.46	49.25
12	Benin	17.34	32.70
13	Ado-Ekiti	23.84	15.13
14	Enugu	35.87	56.22
15	Gombe	22.88	2.24
16	Owerri	34.72	33.07
17	Dutse	22.25	20.26
18	Kaduna	35.08	19.64
19	Kano	0.43	0.19
20	Katsina	26.95	6.37
21	Birnin Kebbi	57.88	58.29
22	Lokoja	48.77	25.73
23	Ilorin	0.00	60.00
24	Ikeja	47.44	31.38
25	Lafia	38.04	55.15
26	Minna	54.86	51.15
27	Abeokuta	20.56	42.32
28	Akure	9.22	35.23
29	Osogbo	15.74	25.13
30	Ibadan	7.06	2.33
31	Jos	42.56	31.63
32	Port Harcourt	38.71	48.24
33	Sokoto	21.14	51.19
34	Jalingo	37.21	34.56
35	Damaturu	49.09	38.99
36	Gusau	12.86	50.83
37	Abuja	20.15	28.69
38	Ijebu Ode	9.98	2.32
39	Ogbomosho	31.79	42.67
40	Ilesa	40.40	29.80
41	Owenna	45.88	10.37
42	Kabba	37.88	30.07
43	Sango Ota	42.76	19.83
44	Zaria	40.61	21.72
45	Ile Ife	59.99	59.99
46	Ikere Ekiti	50.93	49.48
47	Ondo	0.02	30.12
48	Auchi	3.22	48.96
49	Warri	59.99	59.99
50	Oshodi	51.09	31.24
51	Ikotun	11.04	10.84
52	Sagamu	60.00	59.99